



POSTMAN

Supercharging your API
workflow

Luis Becerril

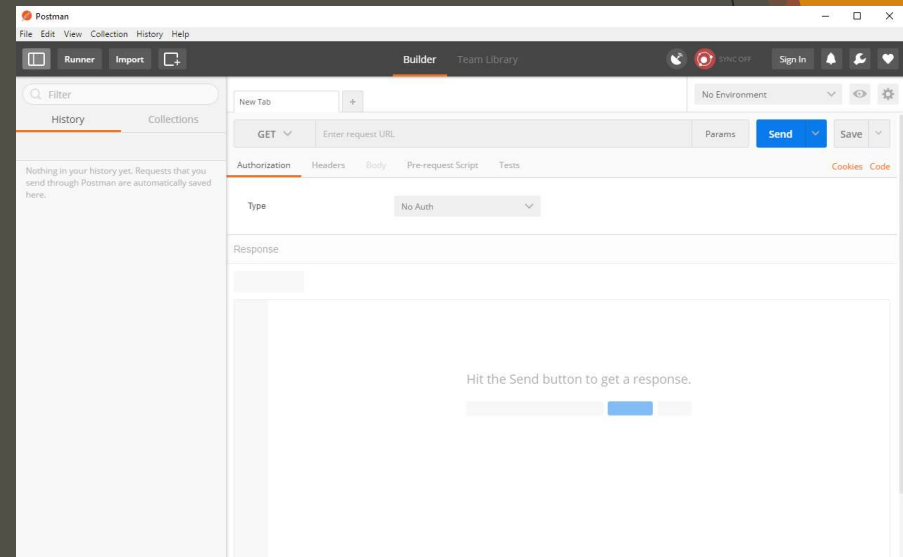
Que es Postman y como me puede ayudar a hacer mi trabajo?

- ▶ GUI que simplifica el desarrollo de API's
- ▶ Ayuda en el desarrollo de API's
- ▶ Ayuda a probar API's
- ▶ Ayuda a documentar API's
- ▶ Ofrece herramientas de colaboracion (paga)



Instalando Postman

- ▶ Dos opciones:
 - ▶ App nativa: viene con add-ons que permiten extender la funcionalidad básica
 - ▶ Chrome app: no es necesario instalar pues corre sobre el navegador
 - ▶ <https://www.getpostman.com/apps>



Enviando un request simple

- ▶ Informacion básica: URL y método
- ▶ Parametros
- ▶ Headers

The screenshot displays a REST client interface with the following components:

- URL input field:** A red circle highlights the text `http://localhost:5000/get?name=abhinav&username=a85`.
- Params:** A red circle highlights the 'Params' button next to the URL field.
- Key-value editor:** A red circle highlights a table used for editing URL parameters. It contains two entries: 'name' with value 'abhinav' and 'username' with value 'a85'. Below the table, the labels 'URL Parameter Key' and 'Value' are visible.
- Buttons:** The interface includes a 'GET' method selector, a 'Send' button, and a save icon.
- Tabs:** At the bottom, there are tabs for 'Authorization', 'Headers (0)', 'Body', 'Pre-request script', and 'Tests'. The 'Authorization' tab is currently selected, showing 'No Auth'.

URL Parameter Key	Value
name	abhinav
username	a85

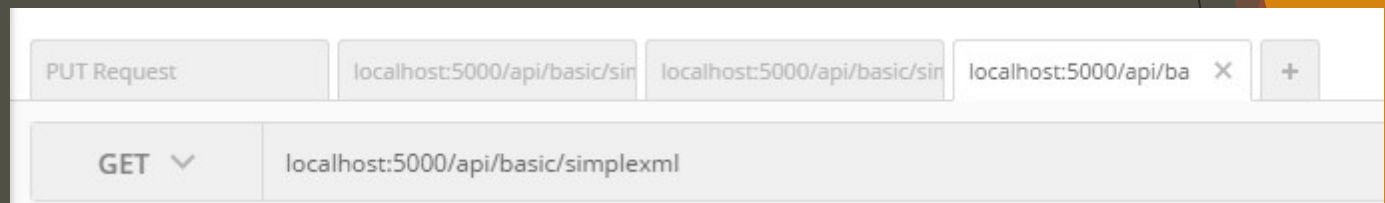
Entendiendo Responses

- ▶ Salvar Response
- ▶ Formatos de response
- ▶ Headers



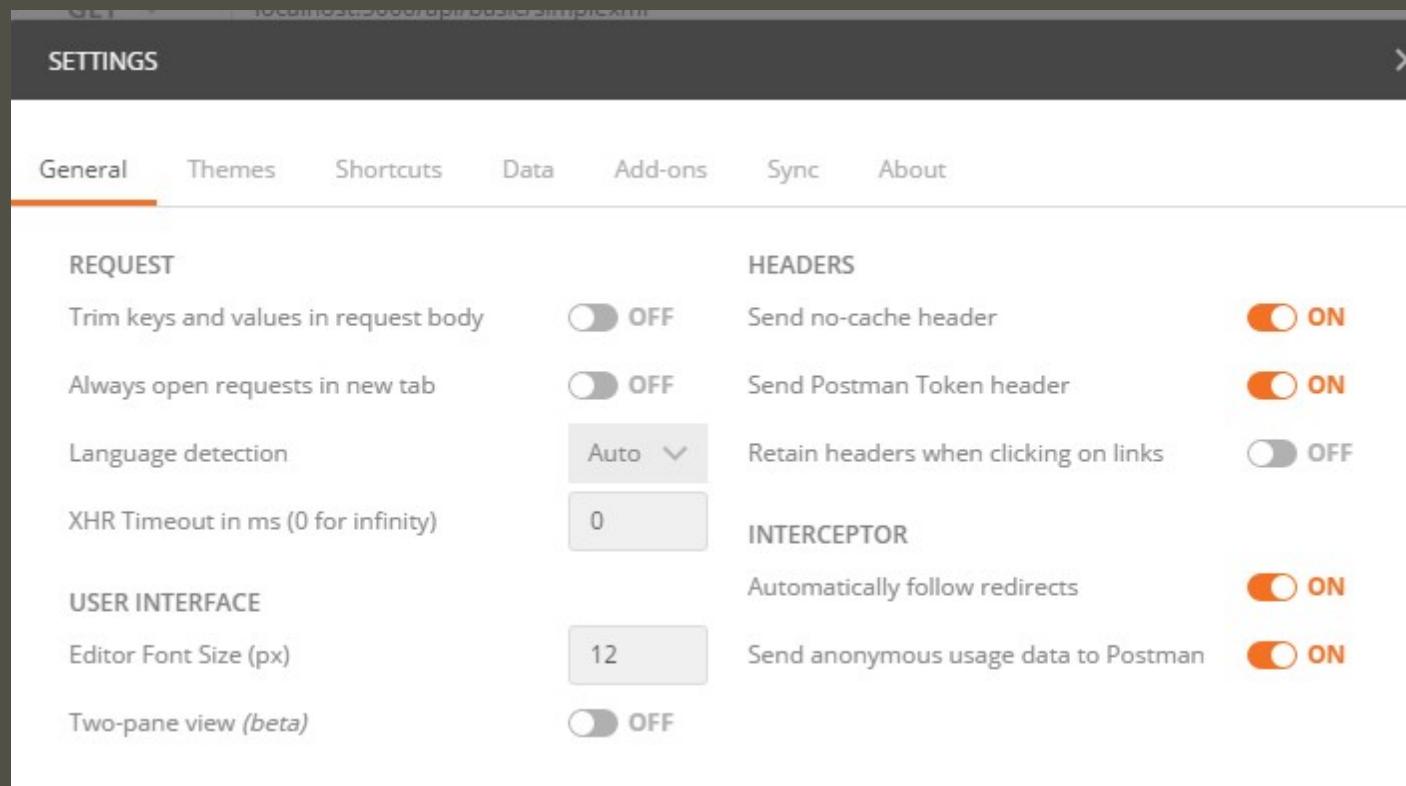
Historia y tabs

- ▶ Permite ver los requests anteriores
- ▶ Permite manejar múltiples requests simultáneos



Herramientas

- ▶ Enviar headers específicos
- ▶ Otras configuraciones



Trabajando con headers

- ▶ Información que el navegador envía trae detalles del request.
 - ▶ Accept, Authorization, Cache-Control, Content-Type, etc.

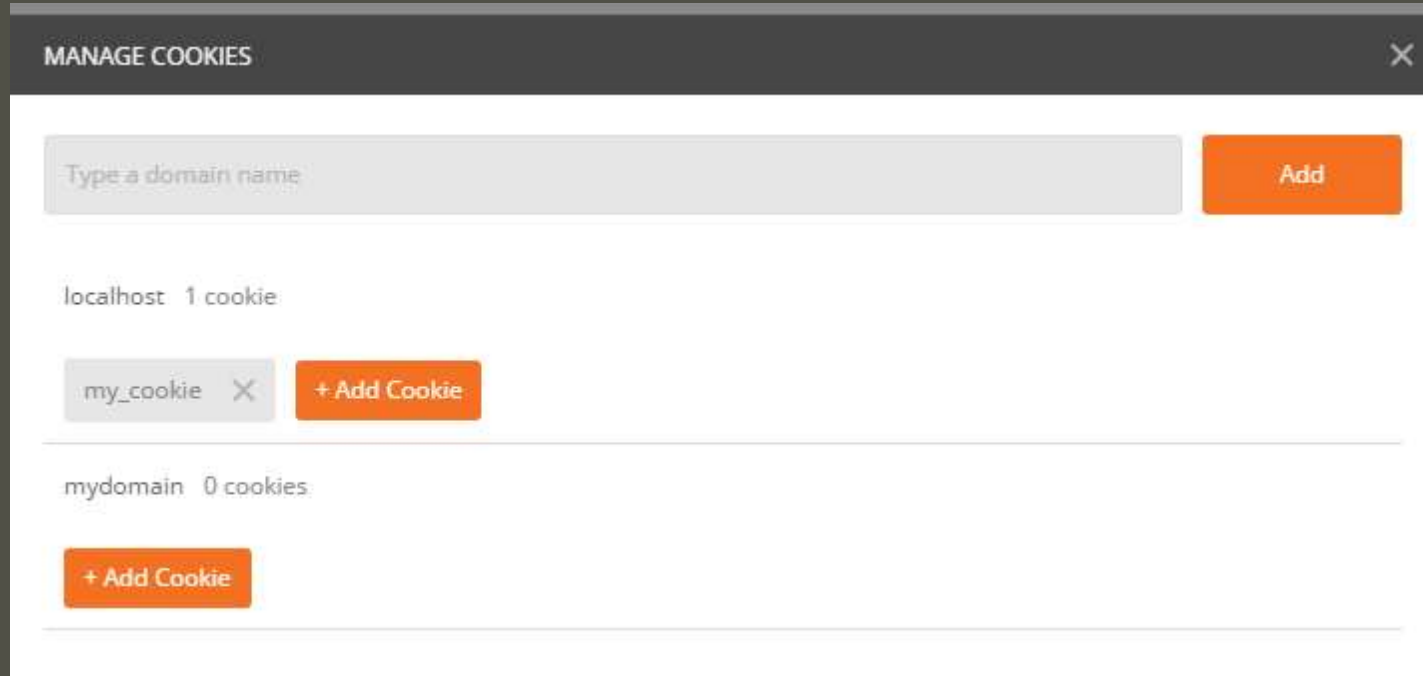


The screenshot shows a web browser's developer tools interface. At the top, there are tabs for 'Pretty', 'Raw', and 'Preview', with 'Pretty' selected. To the right of these tabs is a dropdown menu set to 'JSON' and a 'Save Response' button. The main area displays a JSON object representing the request headers, with line numbers 1 through 11 on the left. The JSON object contains the following key-value pairs:

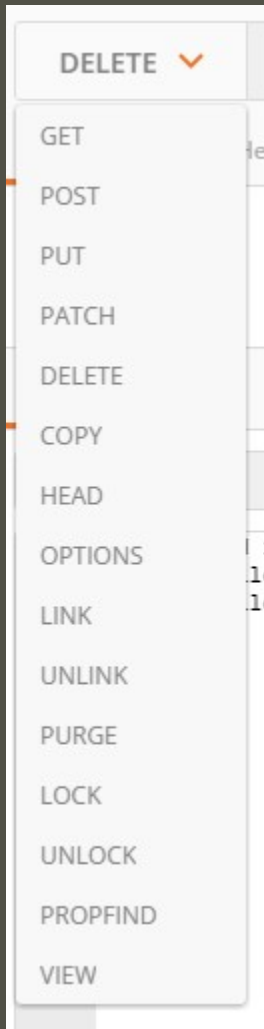
```
1 {  
2   "Cache-Control": "no-cache",  
3   "Connection": "keep-alive",  
4   "Accept": "*/*",  
5   "Accept-Encoding": "gzip, deflate, sdch, br",  
6   "Accept-Language": "en-US,en;q=0.8,es;q=0.6",  
7   "Host": "localhost:5000",  
8   "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36",  
9   "SomeHeader": "my value",  
10  "Postman-Token": "bca67d6e-d941-402e-10e7-a81d067feefb"  
11 }
```


Cookies

- ▶ Son archivos que se guardan en el navegador que permiten guardar información del usuario.
- ▶ Solo en aplicación instalada (no plugin)



Métodos de Request



- ▶ Los métodos indican la acción deseada a realizar en el recurso especificado.
 - ▶ GET, POST, PUT, GET, DELETE



Parámetros de request

- Información extra que se envía con cada petición en el query string.

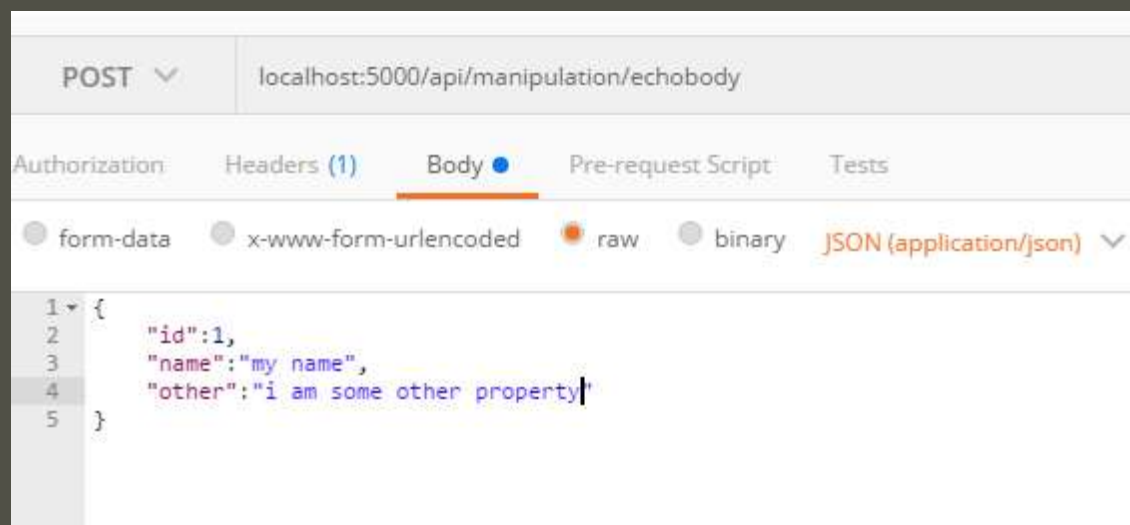
▶ localhost:5000/api/manipulation/echoparameters?someparameter=hello&otherparameter=goodbye&client_id=1

GET ▼	localhost:5000/api/manipulation/echoparameters?someparameter=hello&otherparameter=goodbye&cli	Params
someparameter	hello	≡ ×
otherparameter	goodbye	≡ ×
client_id	1	≡ ×
key	value	



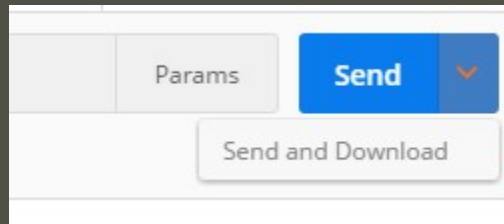
Cuerpo del request

- ▶ El cuerpo del request permite enviar información adicional al request. Permite un grado de personalización más
- ▶ Solo es posible enviar con ciertos métodos de request
- ▶ Es necesario especificar el Content-Type



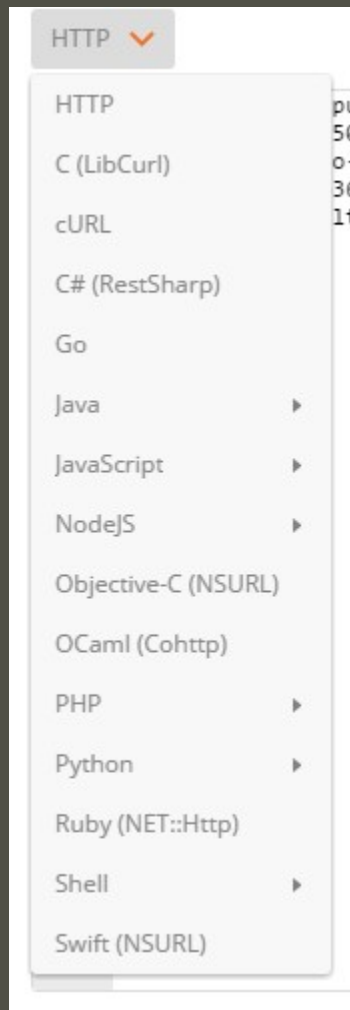
Salvando Responses

- ▶ Es posible salvar las respuestas de los servicios web en un archivo de texto



Exportando un request

- ▶ Es posible salvar un request para usarlo posteriormente
- ▶ Hay varias opciones, la más usada es texto y cURL



Importando un request

- ▶ Es posible importar un request en formato cURL y correrlo en postman

Import a Postman Collection, Environment, data dump, curl command, or a RAML / WADL / Swagger(v1/v2) / Runscope file.

Import File

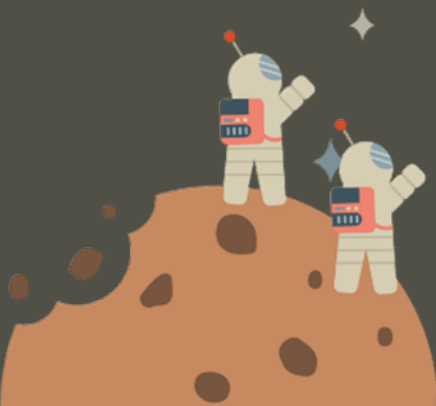
Import Folder

Import From Link

Paste Raw Text

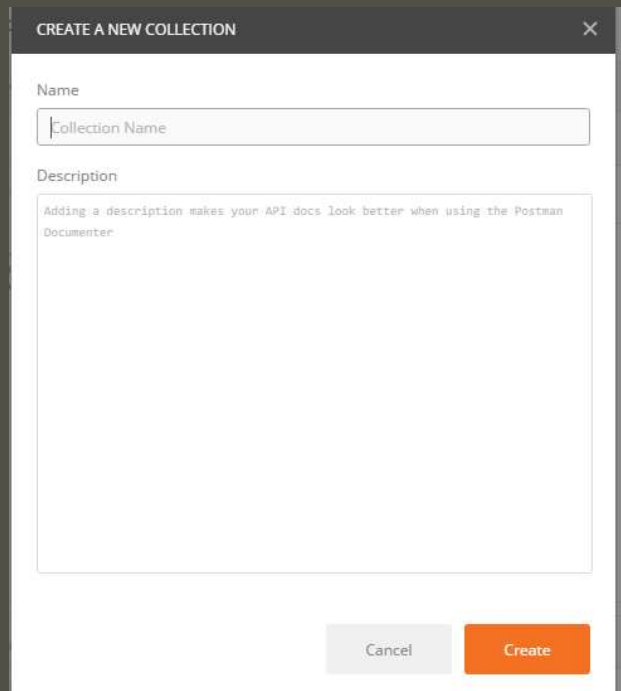
```
curl -X POST -H "Content-Type: application/json" -H  
"Cache-Control: no-cache" -H "Postman-Token: 95b5159d-  
ee5b-bf44-f355-52869f358e71" -d '{  
  "id":1,  
  "name":"my name"  
}' "http://localhost:5000/api/advanced/importexport?  
sampleId=10"
```

Import

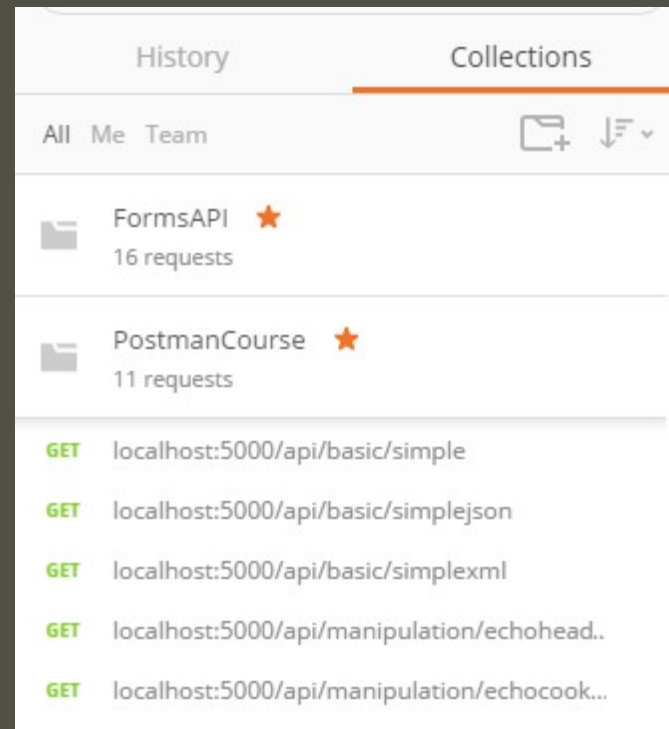


Creando una colección

- ▶ Una colección es una forma de agrupar y salvar requests para su uso posterior
- ▶ Permite añadir documentación del API y facilita el uso compartido de requests.

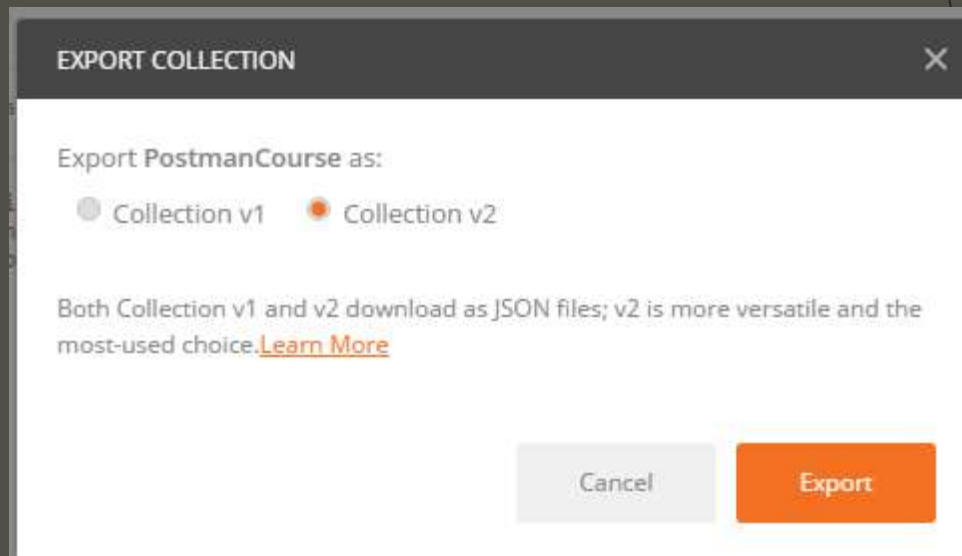
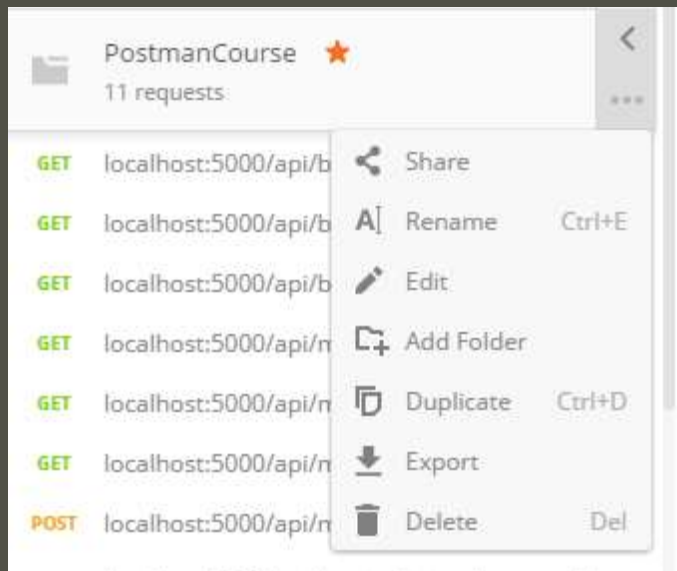


The screenshot shows a modal window titled "CREATE A NEW COLLECTION" with a close button (X) in the top right corner. It contains two input fields: "Name" with a placeholder "Collection Name" and "Description" with a placeholder "Adding a description makes your API docs look better when using the Postman Documenter". At the bottom, there are two buttons: "Cancel" and "Create".



Importando/Exportando una colección

- ▶ Postman facilita la colaboración entre el equipo permitiendo exportar e importar colecciones desde un archivo.



Exportando requests como código

- ▶ Es posible exportar requests como snippets de código en diferentes lenguajes.
- ▶ Facilita el desarrollo y el testing automatizado

C# (RestSharp) ▾

Copy to Clipboard

```
1 var client = new RestClient("http://localhost:5000/api/manipulation/resource/1");
2 var request = new RestRequest(Method.DELETE);
3 request.AddHeader("postman-token", "2e21f5d7-b6d7-068b-6fd2-8b6de1bf7972");
4 request.AddHeader("cache-control", "no-cache");
5 IRestResponse response = client.Execute(request);
```

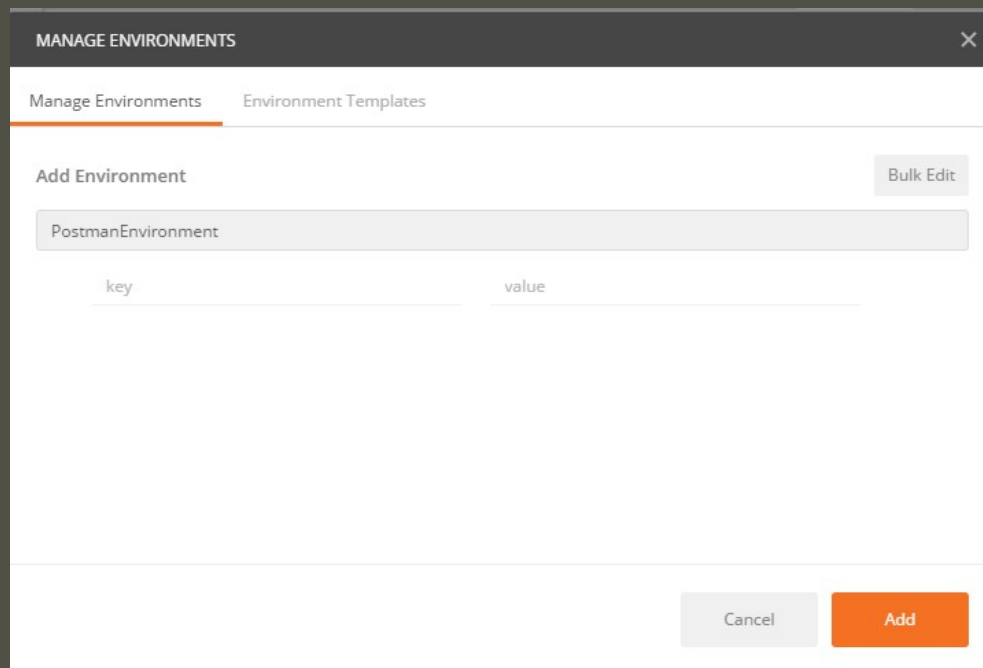
JavaScript Jquery AJAX ▾

Copy to Clipboard

```
1 var settings = {
2   "async": true,
3   "crossDomain": true,
4   "url": "http://localhost:5000/api/manipulation/echoparameters?someparameter=hello&otherparameter=goodbye&client_id=1",
5   "method": "GET",
6   "headers": {
7     "cache-control": "no-cache",
8     "postman-token": "56e7c13d-2b7a-acfe-81a7-3b2ebcb17e3f"
9   }
10 }
11
12 $.ajax(settings).done(function (response) {
13   console.log(response);
14 });
```

Creando un ambiente

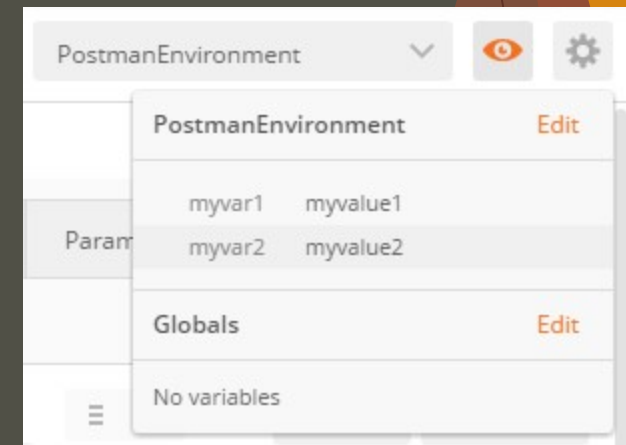
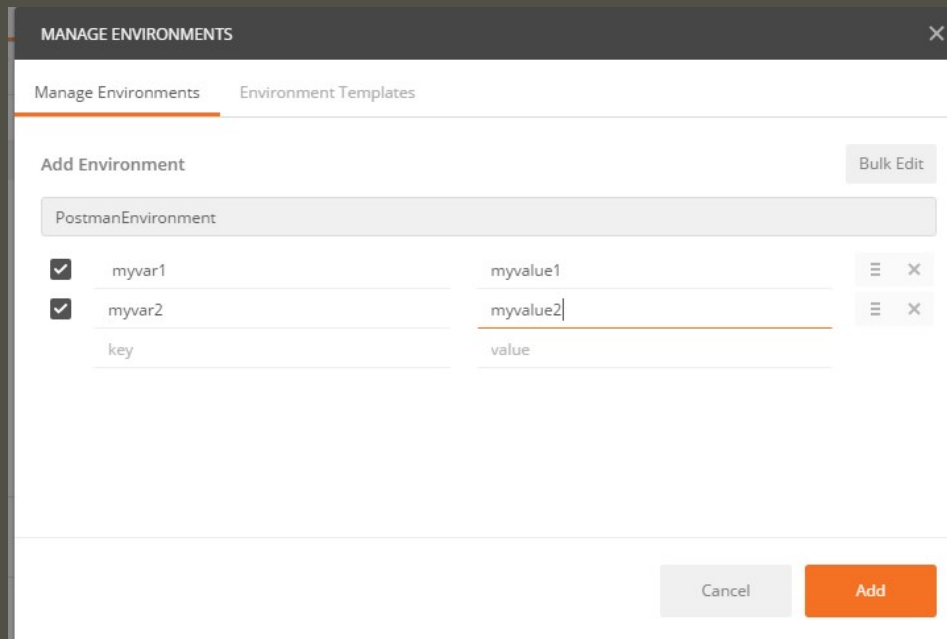
- ▶ Un ambiente es una configuración de diferentes sistemas sobre los cuales se desarrolla un producto (desarrollo, QA, stage, producción)
- ▶ Postman permite emular dichos ambientes para una mejor organización de configuraciones y variables.



The screenshot shows the 'MANAGE ENVIRONMENTS' dialog box in Postman. It has a dark header with the title and a close button. Below the header, there are two tabs: 'Manage Environments' (active) and 'Environment Templates'. The 'Add Environment' section is visible, featuring a 'Bulk Edit' button and a text input field containing 'PostmanEnvironment'. Below this, there are two input fields labeled 'key' and 'value'. At the bottom, there are 'Cancel' and 'Add' buttons.

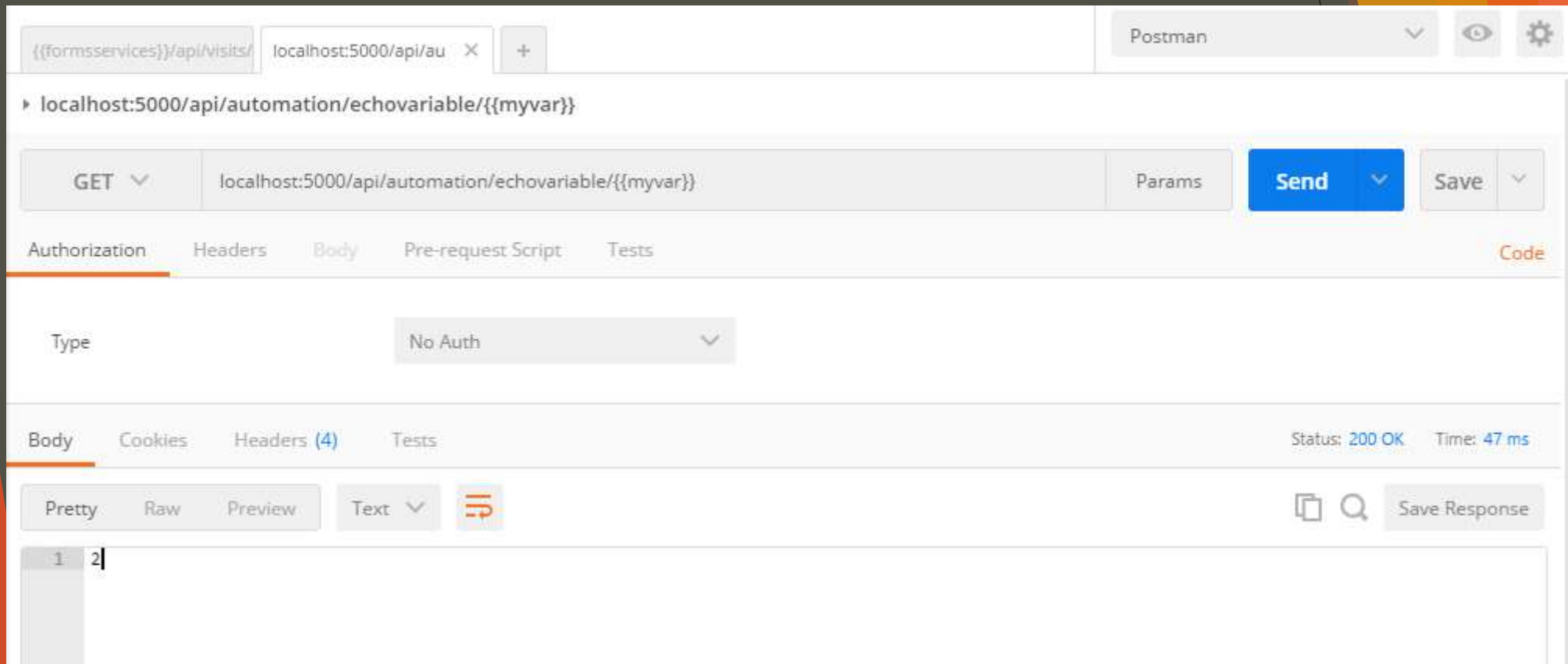
Usando Variables

- ▶ Las variables nos permiten cambiar información de manera más fácil al enviar requests.
- ▶ Existen dos tipos: ambiente y globales
- ▶ Las variables globales se comparten entre todos los ambientes existentes
- ▶ Es posible invocar el valor de las variables con el operador: `{{nombre_de_variable}}`



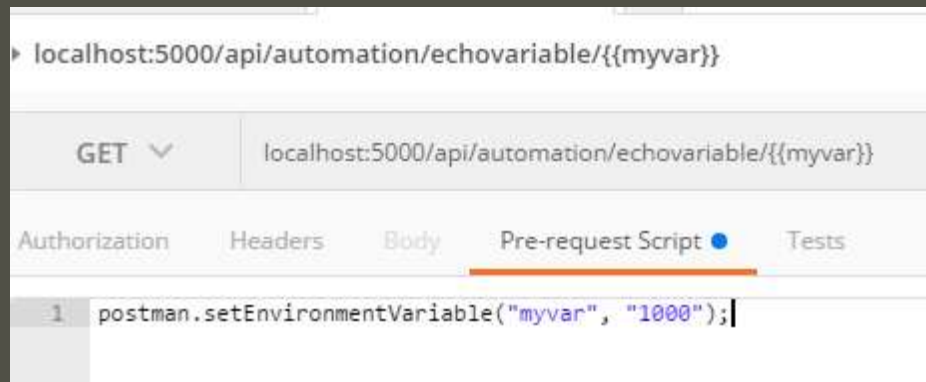
Usando variables

- ▶ Algunos usos posibles son:
 - ▶ URLs de servicios
 - ▶ Parámetros en el request (tokens, etc)



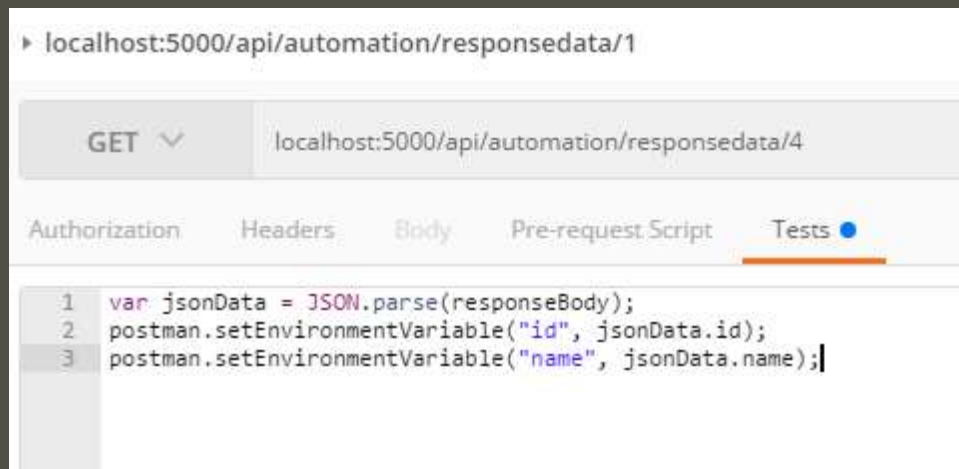
Estableciendo una variable antes del request

- ▶ En requests con variables es útil establecer su valor antes del request
- ▶ Postman ofrece la opción de “Pre-Request Scripts”
- ▶ Se pueden establecer, borrar variables y otros elementos.
- ▶ Acepta comandos en javascript



Extrayendo datos de un response

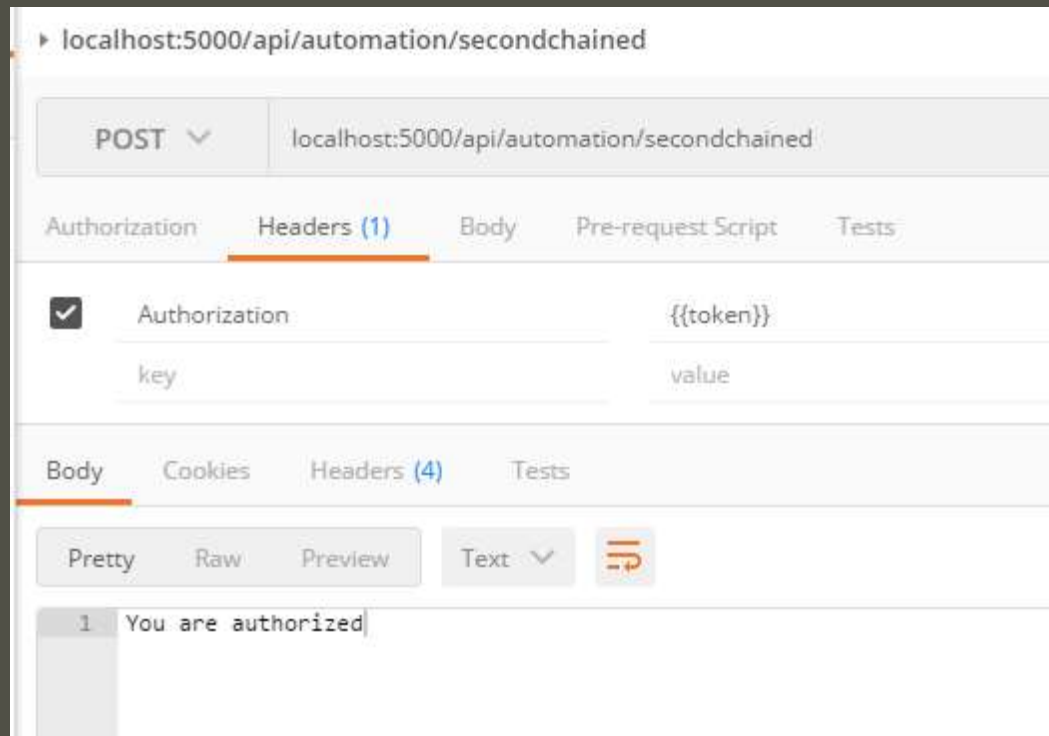
- ▶ Es posible interactuar con la respuesta del servicio web
- ▶ Para esto Postman provee el área de “Tests” los cuales corren después de recibir un response.
- ▶ Se pueden establecer, borrar variables y otros elementos.
- ▶ Acepta comandos en javascript



```
▶ localhost:5000/api/automation/responsedata/1  
GET localhost:5000/api/automation/responsedata/4  
Authorization Headers Body Pre-request Script Tests  
1 var jsonData = JSON.parse(responseBody);  
2 postman.setEnvironmentVariable("id", jsonData.id);  
3 postman.setEnvironmentVariable("name", jsonData.name);
```

Encadenando request manualmente

- ▶ Generalmente para poder enviar un request se depende de datos de un request anterior.
- ▶ Se pueden usar los conocimientos adquiridos para encadenar dos requests, por ejemplo de autenticación



Creando pruebas

- ▶ Postman permite realizar pruebas sobre el reponse de la llamada al servicio web
- ▶ Es posible escribir pruebas con javascript.
- ▶ Las pruebas corren de manera automatica una vez recibido el response.



localhost:5000/api/testing/simpletest

GET localhost:5000/api/testing/simpletest Params Send Save

Authorization Headers Body Pre-request Script Tests Code

```
1 tests["El request fue exitoso"] = responseCode.code === 200;
2 var data = JSON.parse(responseBody);
3 tests["El id esperado es 1"] = data.id === 1;
4 tests["El nombre esperado es hola"] = data.name === "hola";
```

SNIPPETS

- Clear a global variable
- Clear an environment variable
- Response body: Contains string
- Response body: Convert XML body to a JSON Object
- Response body: Is equal to a string
- Response body: JSON value check
- Response headers: Content-Type header check
- Response time is less than 200ms

Body Cookies (4) Headers (4) Tests (1/3) Status: 200 OK Time: 12 ms

All Passed Failed

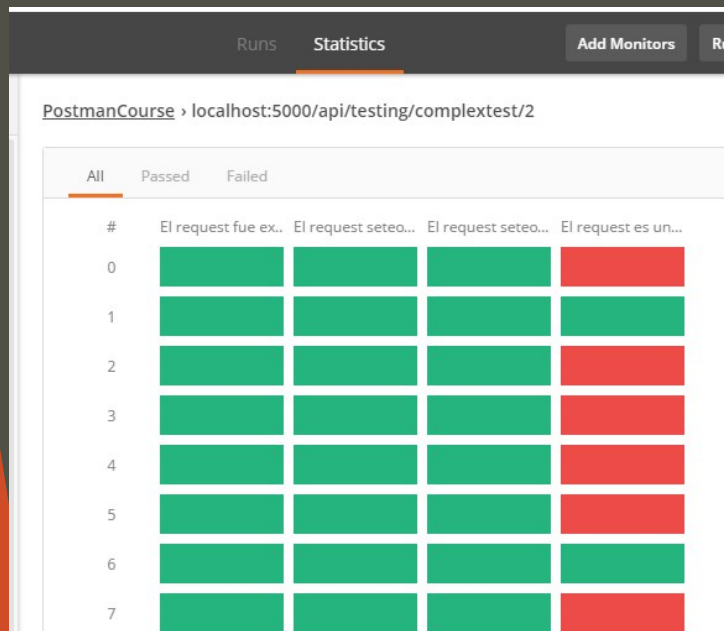
PASS El request fue exitoso

FAIL El id esperado es 1

FAIL El nombre esperado es hola

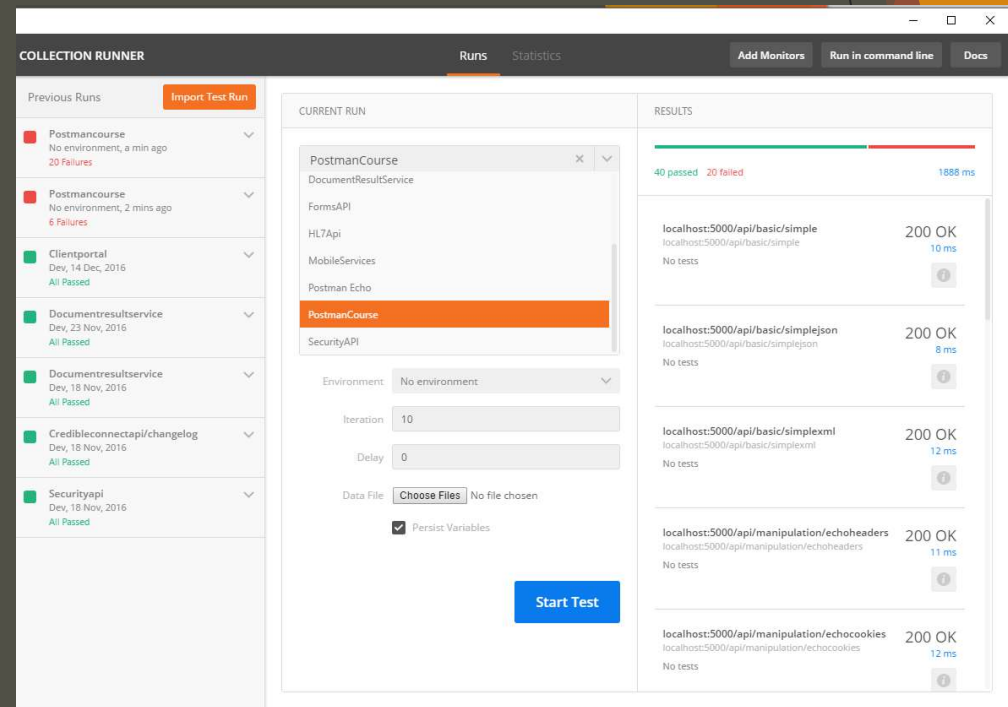
Corriendo una coleccion multiples veces

- ▶ Es posible correr una coleccion definida multiples veces
- ▶ Esto permite realizar pruebas iterativas y probar el flujo de los requests sin problemas.
- ▶ Existe la posibilidad de comparar sets de pruebas y ver estadisticas al respecto



The screenshot shows the 'Statistics' tab in Postman for a collection named 'PostmanCourse' at 'localhost:5000/api/testing/complextest/2'. It displays a table with 8 rows of test results. The first four rows show 'Passed' status, while the last four rows show 'Failed' status. The table has columns for '#', 'El request fue ex...', 'El request seteo...', 'El request seteo...', and 'El request es un...'.

#	El request fue ex...	El request seteo...	El request seteo...	El request es un...
0	Passed	Passed	Passed	Failed
1	Passed	Passed	Passed	Passed
2	Passed	Passed	Passed	Failed
3	Passed	Passed	Passed	Failed
4	Passed	Passed	Passed	Failed
5	Passed	Passed	Passed	Failed
6	Passed	Passed	Passed	Passed
7	Passed	Passed	Passed	Failed



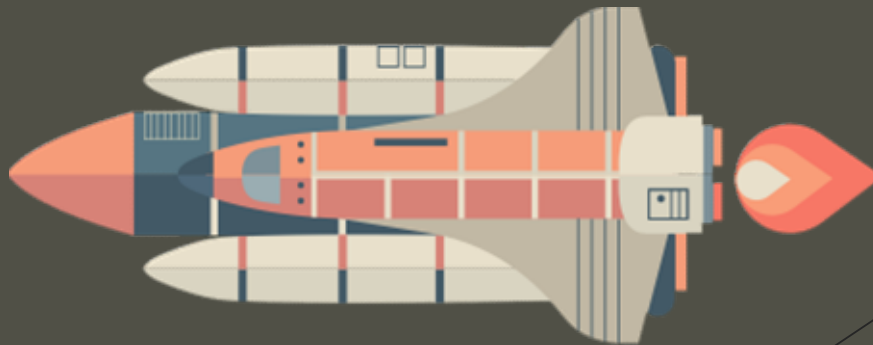
The screenshot shows the 'Collection Runner' interface in Postman. It displays a list of 'Previous Runs' on the left, including 'PostmanCourse' with '20 Failures' and '6 Failures'. The 'CURRENT RUN' section on the right shows the configuration for the 'PostmanCourse' collection, including environment, iteration, delay, and data file settings. The 'RESULTS' section on the far right shows a progress bar and a list of test results, including 'localhost:5000/api/basic/simple' (200 OK, 10 ms) and 'localhost:5000/api/basic/simplejson' (200 OK, 8 ms).

Estableciendo un flujo de trabajo

- ▶ Una vez corriendo una coleccion, los requests se ejecutan en el orden establecido
- ▶ Es posible alterar el orden en el que se ejecutan
- ▶ Esto se logra por medio de la instruccion

```
postman.setNextRequest("request_name");
```

- ▶ Postman se encarga de la redireccion del request. Si no existe el request o no hay redireccion el flujo sigue de manera normal



Ejemplos de testing

```
postman.setEnvironmentVariable("key", "value");
```

```
postman.getEnvironmentVariable("key");
```

```
postman.setGlobalVariable("key", "value");
```

```
var data = JSON.parse(responseBody);  
tests["Your test name"] = data.value === 100;
```

```
tests["Content-Type is present"] = responseHeaders.hasOwnProperty("Content-Type");
```

```
tests["Response time is less than 200ms"] = responseTime < 200;
```

```
tests["Status code is 200"] = responseCode.code === 200;
```

```
tests["Status code name has string"] = responseCode.name.has("Created");
```

```
tests["Body is correct"] = responseBody === "response_body_string";
```

```
var jsonObject = xml2Json(responseBody);
```

```
postman.getGlobalVariable("key");
```

```
tests["Body matches string"] = responseBody.has("string_you_want_to_search");
```